BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D. C. 20024

B69 07032

SUBJECT: UNIVAC 1108 FORTRAN V Version of MIT Conic Subroutines Used in Apollo Guidance Computer - Case 610

DATE: July 10, 1969

FROM: C. O. Guffee
J. C. Gurasich

ABSTRACT

This memorandum contains documentation of the UNIVAC 1108, FORTRAN V version of the conic subroutines as described in Guidance System Operation Plan (GSOP) for program LUMINARY. The conic subroutines form a compatible group of routines which are used extensively by higher level guidance routines in both the Command Module and Lunar Module computers.

All of the conic subroutines have been tested against data obtained from MIT. The MIT data are for tests performed with the Apollo Guidance Computer (AGC) and with a double precision version of the subroutines programmed on an IBM 360 (MAC). The results produced by the UNIVAC 1108 version agree more closely with MAC than do the AGC results.

The conic subroutines are discussed from a user's viewpoint. Possible problem areas are outlined, and a discussion of numerical accuracy and test results are included.

(NASA-CR-106567) UNIVAC 1108 FORTRAN 5 VERSION OF MIT CONIC SUBROUTINES USED IN APOLLO GUIDANCE COMPUTER (Bellcomm, Inc.) 55 p

N79-73142

Unclas
00/60 11585

BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D. C. 20024

B69 07032

SUBJECT: UNIVAC 1108 FORTRAN V Version of MIT Conic Subroutines Used in Apollo Guidance Computer - Case 610

DATE: July 10, 1969

FROM: C. O. Guffee
J. C. Gurasich

MEMORANDUM FOR FILE

I. Introduction

The conic subroutines, as described in Guidance System Operation Plan (GSOP) for program LUMINARY(1), have been programmed in FORTRAN V for the UNIVAC 1108. These subroutines form a compatible group of conic subroutines which are used extensively by higher level guidance routines and programs in both the Command Module and the Lunar Module computers. The conic subroutines are presently being used in subroutines capable of performing the targeting calculations for coelliptic rendezvous maneuvers. The coelliptic rendezvous targeting subroutines are the Apollo on-board routines as described in Reference (1) and include capability for Coelliptic Sequence Initiation (CSI), Constant Differential Altitude (CDH), Transfer Phase Initialization (TPI), and midcourse corrections. The targeting subroutines are being developed jointly by the authors and G. J. Miel (2011), and at this time are in final stage of testing.

A verbal description of the available conic subroutines is contained in Section II followed by Section III with a discussion of the references used in the implementation of the subroutines. Sections IV and V describe the subroutines from a user's viewpoint. Section VI is a discussion of possible logical problem areas of which the user should be aware. Finally, in Section VII test results are presented and possible numerical difficulties are discussed.

II. Conic Subroutines - Description

The conic subroutines can be divided into two groups. The first group contains those subroutines required by higher level guidance subroutines and thus must be called externally. The second group includes the subroutines that do calculations in support of the first group.

The subroutines used by external programs are:

1. Kepler Subroutine: solves for the two-body position and velocity vectors at a terminal position, given the initial position and velocity vectors and a transfer time to the terminal position.

[redacted]

2. Lambert Subroutine: solves for the two-body initial velocity vector, given the initial and terminal position vectors and a desired transfer time.

3. Time-Theta Subroutine: solves for the two-body transfer time, given the initial position and velocity vectors and the true anomaly difference (transfer angle) to the terminal position.

4. Time-Radius Subroutine: solves for the two-body transfer time to a specified radius given the initial position and velocity vectors, the desired radius magnitude, and a flag denoting the upward or downward intersection.

5. Pericenter-Apocenter Subroutine: solves for the two-body pericenter and apocenter altitudes, given the position and velocity vectors for a point on the trajectory.

The subroutines which are required by the above subroutines are:

6. Apsides Subroutine: solves for the two-body radii of apocenter and pericenter and the eccentricity of the trajectory, given the position and velocity vectors for a point on the trajectory.

7. Conic Parameters Subroutine: solves for unit position, unit velocity and unit normal vectors as well as the cotangent of the flight path angle (as measured from the vertical), the normalized semi-latus rectum, and reciprocal of the normalized semi-major axis,* given the position and velocity vectors.

8. Universal Variable Subroutine: solves for the universal variables required to solve for time in the universal form of Kepler's equation. Inputs required are an initial position vector, the cotangent of the flight path angle, the normalized semi-latus rectum, the reciprocal of the normalized semi-major axis, and the central angle from the initial position vector to a final position vector.

9. Kepler Equation Subroutine: solves for the values of the two transcendental functions and for time using the universal form of Kepler's equation, given the universal variables. This subroutine uses a

---

*The semi-latus rectum and semi-major axis are normalized by the magnitude of the initial position vector.

ninth-degree Chebyshev polynomial approximation to the infinite series form for the universal transcendental functions.

10. State Vector Subroutine: solves for the two-body terminal position and velocity vectors, given the universal variables and the solution to Kepler's equation.

Two additional subroutines are also described in the GSOP: the Geometric Parameter Subroutine and the Iterator Subroutine. The Geometric Parameter Subroutine performs calculations which are a subset of the calculations performed by the Conic Parameter Subroutine. The Iterator Subroutine computes the value of the independent variable which drives the error in the dependent variable to zero during the iterations in the Kepler and the Lambert subroutines. In the UNIVAC 1108 formulation by the authors, it was more convenient to build the geometric parameter and iterator into the conic subroutine calculations rather than establish separate subroutines.

Another routine structured like the GSOP model, but considerably more complex than the conic routines is:

11. Initial Velocity Subroutine: computes the initial velocity vector for an integrated trajectory that passes between initial and final position vectors in a specified time.

This subroutine controls a mirror-image iterative targeting process to achieve its answer. It uses alternately the Lambert subroutine and a precision integration package for ballistic flight that includes a full gravity model. An offset target vector used by the Lambert routine is progressively shifted so that the Lambert-computed velocity results in an integrated trajectory that hits the original target vector. The offset is available as an auxiliary output. An input variable specifies the number of iteration cycles, usually three. A zero value will terminate the calculation after the first Lambert solution.

III. Computations Required Within the Subroutines - References

References (1) and (2) were used extensively in writing the FORTRAN version of the subroutines. Reference (1) contains the basic flow charts of the required computations, while Reference (2) contains flow charts that would be required by one who would be programming the on-board computers. Fortunately, Reference (2) relates its nomenclature to the nomenclature as used in the GSOP (Reference (1)) so comparison of the two references is relatively easy. Reference (2) is valuable because details relating to tests performed during the computations, error terminations, corrective action when calculations exceed theoretical limits, and verbal description of the computations are given. In general these tests and required corrective actions are not indicated in the flow diagrams of Reference (1).

Reference (3) was prepared with the intention that it be used together with a symbolic tabulation of the actual computer program. The nomenclature used by this reference is different from that of References (1) and (2); however, Reference (3) does contain a description of each variable which it uses. Two situations arose in which the information contained in References (1) and (2) was either incomplete or incorrect, and in both cases it was possible to produce a working program by interpreting the program listings in Reference (3) along with References (4) and (5).

Finally, References (4), (5) and (6) along with Reference (2) provide the derivations and basic background for understanding the meaning of the computations performed within the conic subroutines.

IV. Arrangement of the Subroutines

A single common block was established for inclusion in each subroutine. Each variable in this common block has exactly the same meaning within all subroutines although all variables are not used within every subroutine. This procedure allows for minimum computation time and minimum storage requirements since a call list is not required when one subroutine calls another subroutine.

As mentioned in a previous section, the Geometric Parameter and the Iterator Subroutines have been built into the routines which call them. These two subroutines do, however, require a call list. This approach appeared reasonable since the built-in form never required more than four or five lines of FORTRAN coding, and each of these subroutines is required by only two of the conic subroutines.

Appendix A identifies all FORTRAN variables used within the conic subroutines along with the nomenclature used in the GSOP. The variables are divided into groups according to their function and are in alphabetical order within each group. All of the conic FORTRAN variables are in a common block /CCØNIC/ which is contained in a PDP deck (described in Appendix B) with entry point QCØNIC* FCØPY. The common block is inserted into the various subroutines at time of compilation by means of the INCLUDE statement.

In addition to the /CCØNIC/ common block, two other common blocks, /CCØN/ and /CSPNT/, are required within the subroutines. /CCØN/ contains conversion constants required within the subroutines and /CSPNT/ contains special print request flags. These common blocks, described in Appendix B, are also compiled into required subroutines by means of the INCLUDE statement. The present arrangement of the three common blocks is only for convenience in using the conic routines in an existing program. The user is free to rearrange the variables into other common blocks as long as all variables are included in the subroutines as required.

In order to use a subroutine via a call from an external program, it is necessary to fill variable values in the common block from input variables. At the conclusion of the computation, values from the common block which are to be output must be stored. To facilitate this a buffer subroutine has been written which contains entry points with associated call lists for each of the required subroutines. Appendix C contains both the subroutine names as they would be called when the variable values are contained within a common block (without a call list) and the subroutine names which would be used externally when data must be carried through a call list. Some subroutines have not been included with a call list name, but the user may add these with the proper calling arguments if their use is required.

The buffer subroutine is listed in Appendix D and comment cards are included to define the call list variables. The buffer subroutine is called MITCØN; however, all calls to this subroutine must be via one of the entry points.

The first entry point shown is ENTRY MITINI(ICBØDY), which is called to initialize certain variables and constants according to the attracting body (presently either Earth or Moon). This entry point must be called one time before using any of the conic subroutines and thereafter a call to this routine is necessary only if the central body should change. The variable values set by this portion of the body are as given in Reference (1) and can be changed by the user, or extended to use the conic subroutine with other attracting bodies.

The remainder of the entry points of MITCØN are documented in the listing of Appendix D. The present form of the call lists are as required by the authors, but freedom exists for the user to increase or decrease these call lists. Appendix E contains a listing of each of the conic subroutines.

A word of caution to the user - in its present form, it is assumed that all input and output vectors are dimensioned four with the magnitude of the vector being the fourth component. It is further assumed that input data via a call list supplies all four components of the vector, and the magnitude of output vectors are always returned through the call list output vectors. If the user either dimensions his vectors by three, or if he does not wish to supply magnitudes of all input vectors, then it is necessary to modify the statements in Subroutine MITCØN which transfers data from call list vectors and the common block vectors. The magnitude can be computed at this time so that all vectors in the common block which are input quantities will contain the magnitude in the fourth position.

Automatic printing of descriptive error messages and of pertinent variable values has been included at necessary points within the subroutines. In addition special printing request flags for printing of the iterations within Kepler and Lambert subroutines are included. These request flags are described in Appendix B under the entry point WSPNT* FCØPY.

A subroutine which prints the current value of all variables contained in the common block /CCØNIC/ has also been written. This printing is initiated by a CALL MITPNT. This subroutine, which is listed in Appendix E, is valuable for diagnostic checks should unexplained problems be encountered in using the conic subroutines. Subroutine MITPNT can be called either from the user's program after a return from a conic subroutine, or by means of an edit at various points within a conic subroutine.

Listings of the conic subroutines are contained in alphabetical order in Appendix F through Appendix P.

V. Supplementary Programs Required

All variable values during diagnostic printing are written by means of an output namelist program. The namelist routine NLØUT is contained in the UNIVAC system and is automatically included whenever its use is required. However, the author uses a special version of NLØUT developed by Miss P. A. Whitlock (2014) which prints six variable values per line of output. Since the system routine prints one to four values per line, a considerable reduction is achieved in output lines of print by using the special version of NLØUT. Instead of using NLØUT one could change to FORTRAN format statements.

Use is made of a package of vector-matrix function routines (Reference (9)) in the FORTRAN coding for the conic subroutines. The user will require either a binary deck of these routines (available from the authors) in order to use the routines in their present form, or replacement of the calls with their equivalent FORTRAN statements.

The Initial Velocity Subroutine has a call to a precision integration subroutine. In the GSOP, the Initial Velocity Subroutine calls the coasting integrating routine, which is an

Encke integration package. The FORTRAN statement CALL EINTEG--- in the Initial Velocity Subroutine calls the authors' version of the GSOP coasting integration package. The user must either remove the Initial Velocity Subroutine or add a precision integration package.

VI. Logic Problems

All of the conic subroutines have been tested extensively and results compared with test data obtained from MIT[7][8]. Problem areas related to programming logic are described below in this section. Section VII discusses computational difficulties and numerical accuracy.

The only logic problem encountered during the test was the iterator logic for the Kepler subroutine. The Kepler subroutine has two features to insure rapid convergence but these two features can also prevent convergence to a correct solution if the user is not aware of the way in which the Kepler subroutine performs the iterations.

The iteration variable in the Kepler subroutine is X. In order to insure rapid convergence, the value of X is confined during the iteration steps to limits of XMIN and XMAX which are computed initially in the Kepler subroutine as XMIN = 0. and XMAX = 2π/SQRT(ALP) or XMAX = SQRT(50./-ALP) depending upon the sign of ALP (the second equation applies to a hyperbola). If the computed values of XMAX exceed a preset value XMAXØ, the subroutine sets XMAX = XMAXØ, the upper limit on the value of X which may occur under normal usage of the Kepler subroutine.*

During the iteration steps, the limits on X are changed according to the direction in which X is to be changed. If the next change in X is to reduce its value then XMAX is set equal to X and then X is reduced for the next iteration step. Likewise, if the next change in X will increase its value, XMIN is set equal to X before X is changed. At no step during the iteration is X allowed to go outside these limits, and the limits are always changed so as to yield a narrower range.

---

*The above limits XMAX and XMIN are for positive transfer time. For negative transfer time the program computes the limits as above and then changes the limits to

XMIN = -XMAX
XMAX = 0.

The remainder of this section is equally applicable for the case negative transfer time.

The initial guess for the value of X and of DELX (the change in X) are computed from a user-supplied value XINIT and the previous solution obtained by the Kepler subroutine given by T21P and XP. The use of the previous solution and an initial guess XINIT provides rapid convergence for the case where repeated calls are made to advance a state vector, as is done with the Encke integration method.* The Kepler subroutine would still converge if XINIT, T21P and XP were all zero; however, extra iterations could be required.

The user must be careful when successive calls to the Kepler subroutine are made with different conics and XINIT, T21P and XP are not zero. The authors found cases where the computed value of DELX on the first iteration was in the wrong direction. This caused the wrong limit on X to be changed with the result that the correct value of X for convergence lay outside the limits [XMAX,XMIN]. On the next iteration, the direction of DELX was computed correctly; however X was now constrained to converge to one of the limits and could not converge to the correct value.

The solution to this problem is to zero T21P, XP and XINIT for each call to the Kepler subroutine except for the case where the subroutine is used in conjunction with the Encke integration method. When used with the Encke integration routine, the values are also zeroed on the initial call and thereafter the subroutine is allowed to work in normal fashion. An alternate solution would be to prevent a change in XMAX or XMIN or the first iteration step. However, since this would involve changing the Kepler subroutine, the authors feel the first approach is the better solution.

The Kepler and Lambert subroutines both use a linear iterator. The new change in X is computed from the previous change in X as

DELX = DELX*(TD-T21)/(T21-T21P).

If the change in T21 is approximately linear with changes in X then there are no problems. However, one test case with a highly eccentric (ECC = 0.9999) eliptical conic required seventy-eight iterations to converge because of the highly non-linear relation of T21 to X. The initial values of T21P, XP and XINIT were all zero for this test. The iterator caused the value of X to oscillate between the two limits, graduately reducing the limits until the correct solution was finally obtained.

---

*See Reference (1) page 5.2-12 for a method of computing XINIT.

It should be noted that the linear iterator will converge to the correct answer but a large number of iterations may result for some conics. Generally, all test cases converged rapidly to a solution; if the user encounters problems with excessive number of iterations it will be necessary to investigate use of a different iteration technique.

If the requested transfer time is larger than one orbital period, the Kepler subroutine subtracts multiples of the orbital period and solves the transfer problem for a time less than one orbital period. A negative desired transfer (TD) time up to one orbit will update the state vector backward in time. However, for larger negative values a wrong answer will result, corresponding to a backward update of exactly one orbit, which except for round-off errors is equivalent to the input vectors. This error was intentionally included to agree with MIT's model.

VII. Test Results and Numerical Difficulties

Test data have been obtained from MIT[7][8] and compared to results from the authors' version of the conic subroutines. MIT ran their test cases with two versions of the programs. The first is the on-board program using the Apollo Guidance Computer (AGC) and the second is an IBM 360 program (MAC).

The AGC is a fifteen-bit fixed-point word machine with one bit reserved for sign.* Most of the computations are performed in double precision which results in a twenty-nine bit, fixed-point word with one bit reserved for sign. Time in the AGC is in double precision and the computations within the DELTII subroutine are performed in triple precision.

The MAC program is in double precision on the IBM 360. The double-precision word on the 360 has sixty-four bits of which nine bits form the exponent and sign, and fifty-five bits used for the fraction. The IBM manual specifies that the double-precision word has seventeen decimal digit accuracy. The 360 is a floating-point machine.

The UNIVAC 1108 version of the conic subroutines has been programmed in single precision. The 1108 word is floating point with nine bits for exponent and sign, and twenty-seven bits for the fraction. This results in eight decimal digit accuracy.

---

*A sixteenth bit is used for parity.

Two distinct computational problems exist within the conic subroutines. The first problem is word length. For example, consider the Kepler subroutine iteration variable X and the resulting Kepler time solution T21. The object is to iterate on X until a value is found for which the resulting solution T21 is equal to (or close to) the desired transfer time TD. For some test cases (particularly a high energy hyperbolic conic) a progression of 1 bit increments in X produces erratic changes in T21. The erratic response is due in part to subtracting two large numbers in the computation of T21 for a hyperbole. The effects are two-fold. First, a change of one digit in X sometimes produces a more than one digit change in T21, which may make it impossible to achieve exact convergence to TD. Second, the derivative of T21 with respect to X, determined by differencing the input and output values, behaves badly for small increments, preventing rapid convergence. Indeed, exceptional cases were observed where the apparent slope had the wrong sign, in violation of the known monotonic function. The solution to this problem is to carry more significant digits by means of double precision.

The second computational problem is that of correctly computing the two transcendental functions CZTA and SZTA. The AGC routine uses a ninth-degree Chebyshev polynominal approximation to the infinite series form for those functions. Even when the Kepler subroutine converges exactly to the desired transfer time TD, the computed final state RT2 and VT2 may be incorrect because of the approximations used to compute CZTA and SZTA. This problem is not a direct consequence of word length but rather of the approximate form used.

Four methods were examined to determine the best way of handling these problems to gain numerical accuracy:

1. Single precision computation using the DELTII subroutine shown in Appendix G.

2. Number 1 with the variables C1, C2, X, DELX, CCØEF (1-10), SCØEF (1-10), T21, ZTA, ALP, CZTA and SZTA as double-precision variables. This results in double-precision computations within the DELTII subroutine.

3. Single-precision computations using the infinite series summation to compute CZTA and SZTA. This subroutine is shown in Appendix Q.

4. Number 3 with the variables C1, C2, X, DELX, ALP, ZTA, T21, CZTA, SZTA, and all variables of DELTII subroutine in double precision. Thus the DELTII subroutine of Appendix Q also performs all computations in double precision.

Typical Kepler test cases involving different types of conics from circular to high energy hyperbolic were tested for all four methods. Each test result was compared to both the corresponding AGC and the MAC results. For most test cases, the AGC and MAC results agree to five or six significant digits. The results of the 1108 tests for each of the above tests can be summarized as follows.

Method 1: The 1108 solutions for circular conics and low-eccentricity elliptical orbits agreed with the MAC results to one or two more significant digits than did the AGC. For high-eccentricity elliptical conics and hyperbolic conics the 1108 solutions were at worst one significant digit less accurate when compared to the MAC than the AGC. One exception was a high-energy hyperbolic conic trajectory for which the 1108 results agreed with MAC to only two significant digits, while the AGC and MAC agreed to five significant digits.

Method 2: Use of double precision did not significantly change the 1108 solutions and resulted in no improvement relative to the MAC and AGC results.

Method 3: With one exception these test results were not significantly different from those of Method 1 and resulted in no improvement in relative accuracy. The exception was the high-energy hyperbolic conic. Test results for this conic were as good as the AGC and for some components of position and velocity were one significant digit better compared with the MAC.

Method 4: Use of double precision and infinite series computation of CZTA and SZTA produced no significant changes in the results of Method 3, and no improvement in the relative answers.

The conclusions drawn with respect to numerical accuracy are:

1. The single precision 1108 conic subroutines provide accurate results if the user expresses RT1, VT1 and PMU as single-precision variables.

2. It would be better to compute CZTA and SZTA using their infinite series form; however, in the majority of the cases the Chebyshev polynomials are adequate.

The Lambert subroutine was tested with the same type of test cases as used for the Kepler subroutine. Only single-precision versions of the Lambert subroutine were tested, but separate tests with both versions of DELTII were used. The results were as described in Methods 1 and 3 above, and the same conclusions with respect to numerical accuracy apply.

Testing of the other conic subroutines resulted in solutions that are consistent with the Kepler and Lambert tests, and the same conclusions with respect to numerical accuracy apply.

C. O. Guffee Jr.

C. O. Guffee

Joan Gurasich

$1025-{}^{\text{COG}}_{\text{JCG}}$-dcs

J. C. Gurasich

Attachments

References

(1) Guidance System Operation Plan for Manned LM Earth Orbital and Lunar Missions Using Program Luminary, (GSOP) Section 5 - Guidance Equations (Revision 1), R-567, November 1968.

(2) Conic Subroutine Flow Charts from MIT Instrumentation Laboratory, Cambridge, Massachusetts, Sundance 302, Document No. FC-1360 dated August 5, 1968 and Document No. FC-1760 dated September 9, 1968.

(3)* Programmed Guidance Equations for Sundance Lunar Module Earth Orbital Program, NAS 9-4810, September 9, 1968.

(4) Marscher, W. F., "A Unified Method of Generating Conic Sections," R-479, MIT/IL, February 1965.

(5) Robertson, W. M., "Explicit Universal Series Solution for the Universal Variable X," MIT/IL, SGA Memo 8'67, May 1967.

(6) Battin, R. H., Astronautical Guidance, McGraw-Hill, Inc., New York, 1964.

(7) Krause, K., Marscher, W. F., Apollo Guidance, Navigation and Control Level I/Level II Test Packages -50, -51, -52, and -53, MIT/IL, September 11, 1967, Revised March 15, 1968.

(8) Computer Printout of test results obtained from W. M. Robertson at MIT/IL.

(9) Guffee, C. O., "Additions to the Vector-Matrix Function Subroutines," Bellcomm Memorandum for File - Case 610, May 7, 1969.

---

*The abstract of this reference specifies that it should not be used as definitive information on the SUNDANCE program; however, the authors found this reference useful in at least two situations in which References (1) and (2) either disagreed or were incomplete.

APPENDIX A

DESCRIPTION OF VARIABLES USED IN CONIC SUBROUTINES

VECTORS

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
EVEC(1-4)	$\underline{e}$	vector directed towards apocenter or pericenter of orbit, defined by RT1 and VT1, with magnitude equal to eccentricity of conic defined by RT1, VT1. The angle from EVEC to RT1 measured in the direction of travel (according to VT1) is between 0. and 180. degrees. EVEC is used by Time-Radius subroutine.
RTT2P(1-4)	$\underline{r}_T'(t_2)$	position vector resulting from precision integration of initial position vector RT1 and initial velocity vector VT1 in Initial Velocity subroutine.
RTT2(1-4)	$\underline{r}_T(t_2)$	a vector used for temporary storage of desired target position vector in Initial Velocity subroutine.
RT1(1-4)	$\underline{r}(t_1)$	initial position vector.
RT2(1-4)	$\underline{r}(t_2)$	terminal position vector.
TSKEP(1-4)	--	temporary storage vector used in Kepler Subroutine.
UEVEC(1-4)	$\underline{u}_e$	unit EVEC.
UN(1-4)	$\underline{u}_N$	unit normal vector in the direction of the angular momentum vector.
URT1(1-4)	$\underline{u}_{r1}$	unit initial position vector.
URT2(1-4)	$\underline{u}_{r2}$	unit terminal position vector.
UVT1(1-4)	$\underline{u}_{v1}$	unit initial velocity vector.
UVT2(1-4)	$\underline{u}_{v2}$	unit terminal velocity vector.
VTT2P(1-4)	$\underline{v}_T'(t_2)$	velocity vector associated with RTT2P.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
VT1(1-4)	$\underline{v}(t_1)$	initial velocity vector.
VT2(1-4)	$\underline{v}(t_2)$	terminal velocity vector.

GENERAL VARIABLES

GV1(1-4) GV2(1-4)	-- --	vectors used in intermediate calculations as required.
ISTATE	--	an indicator carried into Initial Velocity subroutine for call to Encke integration package. This indicates to the integration package which gravity model should be used.
TIME1	--	a variable carried into Initial Velocity subroutine for call to Encke integration package. TIME1 is the time since zero time and is used in conjunction with computations requiring ephemeris data within the integration subroutine.
TS1 TS2 TS3 TS4	-- -- -- --	variables used in intermediate calculations for temporary storage.

CONIC VARIABLES

ALP	α	reciprocal of semi-major axis (negative for hyperbolas).
ALPN	α_N	ratio of magnitude of initial position vector to semi-major axis (negative for hyperbolas).
CØSF	--	cosine of F.
CØSF2	--	(cosine of F) **2.
CØTTØ2	--	cotangent of THETA/2.
CTHETA	--	cosine of THETA.
ECC	e	eccentricity.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
F	f	angle from apocenter or pericenter to RT2 measured in direction of motion so that F is between 0. and 180. degrees.
GAM	γ	inertial flight path angle measured from vertical ($0 \leq \gamma \leq 180$ deg).
HA	h_A	altitude at apocenter.
HP	h_P	altitude at pericenter.
PN	p_N	ratio of semi-latus rectum to magnitude of initial position vector.
RA	r_A	radius of apocenter.
RP	r_P	radius of pericenter.
SINF	--	sine of F.
SINGAM	--	sine of GAM.
SQRPN	--	square root of PN.
STHETA	--	sine of THETA.
THETA	θ	true anomaly difference between RT1 and RT2.
TP	t_P	period of conic as defined by RT1 and VT1.

CONSTANTS

CCØEF(1-10)	--	contains the Chebyshev coefficients for the 9th degree polynomial approximation to the C-transcendental function's infinite series solution.
CØEFGX(1-6)	--	contains the Chebyshev coefficients for the 6th degree polynomial approximation to the infinite series, for evaluating the value of XN in the Universal Variable subroutine.

Nomenclature		Description
Bellcomm	GSOP	
CØTMN	--	value of cotangent of 1 deg 47.5 min. Used in Lambert subroutine to limit the initial guess as to the value of CØTMIN.
CØTMX	--	value of cotangent 178 deg 72.5 min. Used in Lambert subroutine to limit the initial guess as to the value of CØTMAX.
IMØØN	PC	=1, Moon is attracting body, =0, Earth is attracting body.
MITKEP	--	maximum number of iterations allowed in Kepler subroutine.
MITLAM	--	maximum number of iterations allowed in Lambert subroutine.
PMU	μ	product of universal gravitational constant and mass of the primary attracting body.
RB	r_b	radius of attracting body.
RMAX	r_{MAX}	the radius of apocenter is not defined for parabola or hyperbola so it is set to RMAX in Apsides subroutine.
SCØEF(1-10)	--	contains the Chebyshev coefficients for the 9th degree polynomial approximation to the S-transcendental function's infinite series solution.
SG	s_G	a value of either +1. or -1. according to whether the true anomaly difference between RT1 is respectively less than or greater than 180 degrees.
SRR	$s_{\dot{R}}$	a value of either +1. or -1. according to whether the desired radial velocity at RT2 is respectively plus or minus in Time-Radius subroutine.
SQRPMU	--	square root of PMU.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
XMAXØ	x_{MAXO}	absolute upper bound on Kepler iteration variable X set according to the attracting body.

FLAGS

Bellcomm	GSOP	Description
IF1	f_1	a switch set to 0 or 1 according to whether a guess of cot γ is available or not (=0, guess is available).
IF2	f_2	a switch set to 0 or 1 according to whether Lambert should determine $\underline{u}_N$ from $\underline{r}(t_1)$ and $\underline{r}(t_2)$ or $\underline{u}_N$ is an input.
IF3	f_3	a tag set to 0 or 1 according to whether the iterator should use the "Regula Falsi" or bias method.
IF4	f_4	a flag set to 0 or 1 according to whether the iterator is to act as a first order of a second order iterator.
IF5	f_5	a flag set to 0 or 1 according to whether a feasible solution exists or not.
IF6	f_6	a switch set to 0 or 1 according to whether or not the new state vector is to be an additional output requirement of the Time-Theta or Time-Radius problems.
IF7	f_7	a flag set to 1 if the inputs require that the conic trajectory must close through infinity.
IF8	f_8	a flag set to 1 if the Time-Radius problem was solved for pericenter or apocenter instead of $r(t_2)$.
IF9	f_9	a flag set to 1 if the input to the Time-Radius subroutine produces an e less than 2^{-18}.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
IFCØGA	f_γ	=1, CØTGAM not in range (1° 47.5', 175° 12.5') =0, CØTGAM is within range
IFN1	n_1	=1, Lambert returns VT1 and CØTGAM =0, Lambert returns VT1, VT2 and CØTGAM
IFW	f_ω	a flag set to 1 in the Universal Variable subroutine if θ is nearly less than 360°, in which case the x_N corresponding to 360°-θ is calculated and subtracted from the x_N corresponding to 360° exactly.
IPKEP	--	flag used to test for required printing of an iteration step in Kepler subroutine.
IPLAM	--	flag used to test for required printing of an iteration step in Lambert subroutine.
IPTKEP	--	flag set to 1 if Kepler subroutine does not converge within maximum number of iterations. The subroutine then reinitializes itself and prints the iterations as they are performed.
IPTLAM	--	flag set to 1 if Lambert subroutine does not converge within maximum number of iterations. The subroutine then reinitializes itself and prints the iterations as they are performed.

ITERATION VARIABLES

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
A	--	temporary iteration variable used in Universal Variable subroutine.
CK	k	a fraction of the full value of the full range of the independent variable which determines the increment of the independent variable on the first pass through the iterator in Lambert subroutine.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
CØSGAM	$\cos\gamma$	cosine of GAM.
CØTGAM	$\cot\gamma$	contangent of GAM - this is the iteration variable in the Lambert subroutine.
CØTMAX	$\cot\gamma_{max}$	upper limit for the value of CØTGAM during iterations in Lambert subroutine.
CØTMIN	$\cot\gamma_{min}$	lower limit for the value of CØTGAM during iterations in Lambert subrotine.
CZTA	$c(\xi)$	value of the C-transcendental function (with argument ZTA) as used in the universal form of Kepler's equation.
C1	c_1	a constant used in computing T21 in the universal form of Kepler's equation. C1 is computed as either (RT1 (dot) UT1/SQRPMU) in Kepler subroutine or as SQRT(PN*RT1(4))*CØTGAM in Universal variable subroutine. These are equivalent computations.
C2	c_2	a constant used in computing T21 in the universal form of Kepler's equation. C2 is computed as RT1(4)*VT1(4)**2/SQRPMU -1. in Kepler's subroutine or as 1.-ALPN in Universal Variable subroutine. These are equivalent computations.
C3	c_3	a constant computed as RTL(4)*VT1(4)**2/PMU.
DCØTG	$\Delta_{\cot\gamma}$	increment in X which will produce a smaller value in TERR. DELX is used to change the iteration variable CØTGAM in Lambert subroutine.
DELX	Δ_X	increment in X which will produce a smaller value in TERR. DELX is used to change the iteration variable X in Kepler's subroutine.
EPSK EPSL	ε_t	fraction which when multiplied by the desired transfer time will yield the error allowed in the solutions within Kepler and Lambert subroutines. EPSK is used in Kepler subroutine and EPSL is used in Lambert subroutine.

Nomenclature		Description
Bellcomm	GSOP	
EPSINV	ε	cone angle of a cone whose vertex is the coordinate origin and whose axis is the 180 degree transfer direction (i.e., the negative initial position vector). This is used in Initial Velocity subroutine to determine if transfer is too close to 180 degrees and hence the target vector must be rotated into the orbital plane.
EPSKEP	--	absolute value product of EPSK and TD computed once in Kepler subroutine to use in test for convergence.
EPSLAM	--	absolute value produce of EPSL and TD21 computed once in Lambert subroutine to use in test for convergence.
IDELT	--	iteration counter in Kepler Equation subroutine.
ITGETX	--	iteration counter in Universal Variable subroutine.
ITKEP	i	iteration counter in Kepler subroutine.
ITLAM	i	iteration counter in Lambert subroutine.
N1	n_1	number of iterations to be used in calculation the offset target vector in Initial Velocity subroutine.
N2	n_2	iteration counter in Universal Variable subroutine.
OMEGA	ω	cosine of EPSINV.
P1	p_1	constant used within Lambert subroutine computed one time only as CTHETA-ZLAM.
P2	p_2	constant used within Lambert subroutine computed one time only as CTHETA-ZLAM.
SZTA	$s(\xi)$	value of the S-transcendental function (with argument ZTA) as used in the universal form of Kepler's equation.

Nomenclature		Description
Bellcomm	GSOP	
TD	t_D	desired transfer time through which the conic update of the state vector is to be made (used in call to Kepler subroutine).
TD21	t_{D21}	desired transfer time to traverse from RT1 to RT2 (used in call to Lambert subroutine).
TERR	t_{ERR}	error between desired transfer time (either TD or TD21) and solution given by Kepler's equation (T21) for current value of iteration variable.
TR	t_R	integral periods subtracted from TD to produce a TD less than TP in Kepler's subroutine.
T21	t_{21}	transfer time as given by solution to universal form of Kepler's equation.
T21P	t'_{21}	transfer time corresponding to the previous solution to Kepler's equation associated with iteration variable value XP.
W1	w_1	temporary iteration variable used in Universal variable subroutine.
W2	w_2	temporary iteration variable used in Universal variable subroutine.
W3	w_3	temporary iteration variable used in Universal variable subroutine.
X	x	a universal conic parameter equal to the ratio of eccentric anomaly difference to SQRT(ALP) for an ellipse or the ratio of the hyperbolic analogue of eccentric anomaly to SQRT(-ALP) for a hyperbola.
XINIT	x_{INIT}	initial guess for value of X for call to Kepler subroutine.
XMAX	x_{MAX}	upper limit for value of X during iterations in Kepler subroutine. A new guess for X is not allowed to exceed MAX.

Nomenclature: Bellcomm	Nomenclature: GSOP	Description
XMIN	x_{MIN}	lower limit for value of X during iterations in Kepler subroutine. A new guess for X is not allowed to be lower than XMIN.
XN	x_N	ratio of X to magnitude of initial position vector (RT1(4)).
XP	x'	value of X used for previous Kepler's equation solution (see TP).
XR	x_R	value of X associated with TR.
X2	--	X**2
X2CZTA	--	X2*CZTA
X3	--	X**3
ZLAM	λ	ratio of magnitude of initial position vector to final position vector (RT1(4)/RT2(4)).
ZTA	ξ	argument of transcendental function in the universal form of Kepler's equation.

APPENDIX B

COMMON BLOCKS FOR CONIC SUBROUTINES

```
QCONIC* FCOPY
C
C       THIS COMMON BLOCK IS USED INTERNAL TO THE MIT CONIC SUBROUTINES
C
        COMMON/CCONIC/  RT1(4),        VT1(4),        RT2(4),
     .  VT2(4),         URT1(4),       URT2(4),       UVT1(4),
     .  UVT2(4),        GV1(4),        GV2(4),        UN(4),
     .  GAM,            TS1,           TS2,
     .  TS3,            TS4,           EPS,
     .  SG,             F,             X,
     .  PMU,            SGRPMU,        XMAXO,         RMAX,
     .  RB,             COTMX,         COTMN,         IMOON,
     .  EPSK,           EPSL,
     .  RP,             RA,            ECC,
     .  T21,            SZTA,          X2CZTA,        CZTA,
     .  CCOEF(10),      SCOEF(10),     W2,            W3,
     .  IDELT,
     .  XN,             X2,            ZTA,           COTTO2,
     .  W1,             IFW,           ITGETX,        COEFGX(6),
     .  A,
     .  ALP,            XP,            TD21,          TD,
     .  TERR,           DELX,          XMAX,          XMIN,
     .  XINIT,          XR,            TR,            TP,
     .  CK,             C1,            C2,            EPSKEP,
     .  TSKEP(4),       MITKEP,        ITKEP,         IPTKEP,
     .  IPKEP,          IF4,
     .  T21P,           DCOTG,         MITLAM,        ZLAM,
     .  COTMAX,         COTMIN,        P1,            P2,
     .  EPSLAM,         IF1,           IF2,           IF3,
     .  IF5,            IFN1,          ITLAM,         IPLAM,
     .  IPTLAM,
     .  X3,
     .  ALPN,           PN,            COSGAM,        SINGAM,
     .  COTGAM,         C3,            SQRPN,
     .  HA,             HP,
     .  SRR,            EVEC(4),       UEVEC(4),      IF8,
     .  IF9,            COSF,          COSF2,         SINF,
     .  THETA,          STHETA,        CTHETA,        IF6,
     .  IF7,            IFCOGA,
     .  N1,             N2,            OMEGA,         EPSINV,
     .  ISTATE,         TIME1,         RTT2P(4),      VTT2P(4),
     .  RTT2(4)
C
       END
```

```
WCON* FCOPY
C
C **** COMMON BLOCK FOR UNIVERSAL CONSTANTS AND CONVERSION FACTORS.
C    DTOR         CONVERSION FROM DEGREES TO RADIANS (0.0174532925)
C    RTOD         CONVERSION FROM RADIANS TO DEGREES (57.2957796)
C    HALFPI       PI/2.(1.57079633)
C    PI           RADIANS IN HALF CIRCLE (3.14159265)
C    TWOPI        2.*PI (6.28318531)
C    HRDAY        HOURS IN A DAY (24.)
C    SECDAY       NUMBER OF SECONDS IN A DAY (86400.)
C    SECHR        NUMBER OF SECONDS IN AN HOUR (3600.)
C    SIXTY        THE NUMBER SIXTY (60.)
C    DEGCIR       NUMBER OF DEGREES IN A CIRCLE (360.)
C    FTNM         NUMBER OF FEET IN A NAUTICAL MILE (6076.1155)
C    FTMT         NUMBER OF FEET IN A METER (1./0.3048)
C    BIGNO        A BIG NUMBER (1.E30)
C    SMLNO        A SMALL NUMBER (1.E-37)
C    GZERO        ACCELERATION OF GRAVITY AT SURFACE. THIS CONSTANT IS USED
C                 TO CONVERT WEIGHT TO MASS WITHIN THE PROGRAM.
C
      END
C
C
QCON* FCOPY
C
C      THIS COMMON BLOCK CONTAINS UNIVERSAL CONSTANTS AND CONVERSION
C      FACTORS REQUIRED BY THE PROGRAM.
C
       COMMON/CCON/    DTOR,          RTOD,          HALFPI,
     . PI,             TWOPI,         HRDAY,         SECDAY,
     . SECHR,          SIXTY,         DEGCIR,        FTNM,
     . FTMT,           BIGNO,         SMLNO,         GZERO
C
       DATA DTOR/0.0174532925/, RTOD/57.2957796/, HALFPI/1.57079633/,
     . PI/3.14159265/, TWOPI/6.28318531/, HRDAY/24./, SECDAY/86400./,
     . SECHR/60./, DEGCIR/360./, FTNM/6076.1155/, FTMT/3.2808399/,
     . BIGNO/1.E30/, SMLNO/1.E-37/, GZERO/32.174048/
C
      END
C
```

```
*SPNT* FCOPY
C
C       THIS COMMON BLOCK CONTAINS SPECIAL PRINT INDICATORS AS REQUIRED
C       FOR SPECIFYING PRINTING WITHIN THE SUBROUTINES.
C    JPKEP       =N,PRINT EVERY N-TH ITERATION IN SUBROUTINE KEPMIT,
C                =0,DO NOT PRINT ITERATIONS IN KEPMIT. SEE 'JPTKEP'.
C    JPTKEP      =0,PRINT STARTING VALUES AND SOLUTION IN SUBROUTINE
C                KEPMIT IF SUBROUTINE DOES NOT CONVERGE WITHIN MAXIMUM
C                ALLOWABLE NUMBER OF ITERATIONS.  =N,PRINT EVERY N-TH
C                ITERATION IN KEPMIT IF SUBROUTINE DOES NOT CONVERGE TO
C                AN ANSWER
C    JPLAM       (SIMILAR TO JPKEP EXCEPT FOR SUBROUTINE LAMMIT)
C    JPTLAM      (SIMILAR TO JPTKEP EXCEPT FOR SUBROUTINE LAMMIT)
C
      END
C
C
CSPNT* FCOPY
C
       COMMON/CSPNT/
     . JPKEP,           JPTKEP,           JPLAM,            JPTLAM,
C
      END
C
```

APPENDIX C

FORTRAN NAMES FOR THE CONIC SUBROUTINES

Conic Subroutine	Calling Name for Data via Common Block	Calling Name for Data via Call List
Kepler	KEPMII	KEPMIT
Lambert	LAMMII	LAMMIT
Time-Radius	TRADI	TRAD
Time-Theta	TTHETI	TTHETA
Apsides	APSIDI	--
Conic Parameter	PARAMI	--
Universal Variable	GETXI	--
Kepler Equation	DELTII	--
State Vector	NEWSTI	--
Initial Velocity	INITVI	INITV
Pericenter-Apocenter	PERAPI	PERAPØ

APPENDIX D

SUBROUTINE MITCON - THE BUFFER SUBROUTINE

```
            SUBROUTINE MITCON
C     THIS SUBROUTINE CONTAINS THE CALL LISTS FOR THE MIT CONIC SUROUTINES.
C     ACTUAL CALLS TO THE CONIC SUBROUTINES ARE MADE BY THIS SUBROUTINE
C     AND VARIABLES ARE CARRIED VIA COMMON TO THE CONIC SUBROUTINES.
            INCLUDE QCONIC
            INCLUDE QCON
C
C     ICBODY   INDICATOR =1,MOON IS CENTRAL BODY,  =2, EARTH IS CENTRAL
C              BODY
C     R1(1-4)  INITIAL POSITION
C     V1(1-4)  INITIAL VELOCITY
C     R2(1-4)  FINAL POSITION
C     V2(1-4)  FINAL VELOCITY
C     V1R(1-4) VELOCITY REQUIRED AT R1 TO ARRIVE AT R2
C     TDES     DESIRED TRANSFER TIME FOR R1 TO R2
C     ZTHETA   TRUE ANOMALY DIFFERENCE BETWEEN R1 AND R2
C     MIFG     INDICATOR =0 OR 1 ACCORDING TO WHETHER OR NOT THE NEW
C              STATE SHOULD BE COMPUTED IN TIME-THETA SUBROUTINE
C     ZALPN    RATIO OF MAGNITUDE OF INITIAL POSITION TO SEMI-MAJOR AXIS
C     ZT21     REQUIRED TRANSFER TIME FROM R1 TO R2
C     R2G      MAGNITUDE OF FINAL POSITION
C     ZHP      HEIGHT ABOVE PLANET AT PERICENTER
C     ZPN      RATIO OF SEMI-LATUS RECTUM TO MAGNITUDE OF INITIAL
C              POSITION
C     PMUCB    PRODUCT OF UNIVERSAL GRAVITATIONAL CONSTANT AND MASS OF
C              PRIMARY ATTRACTING BODY
C     ZXINIT   INITIAL GUESS AS TO VALUE OF X FOR CALL TO KEPLER
C     GCOTG    INITIAL GUESS TO VALUE OF COTGAM  FOR CALL TO LAMBERT
C              (MUST BE ZERO IF A GUESS IS NOT AVAILABLE-- THE SOLUTION
C              OBTAINED IN LAMBERT IS RETURNED)
C     ZSRR     INPUT EQUAL TO +1. OR -1. DEPENDING UPON WHETHER RT2 AS
C              DETERMINED BY TIME-RADIUS SUBROUTINE  HAS A
C              POSITIVE OR NEGATIVE RADIAL RATE
C     ZECC     ECCENTRICITY OF ORBIT RT1,VT1 (OUTPUT OF CONIC SUBROUTINE)
C     IVC      INDICATOR CARRIED INTO INITV SUBROUTINE FOR USE IN CALL TO
C              PRECISION INTEGRATION PACKAGE
C     NOSETZ   NUMBER OF OFFSET ITERATIONS TO BE PERFORMED IN INITV SUBROUTINE
C     R2OSET   THE OFFSET TARGET VECTOR  OUTPUTTED BY INITV
C     ZEPS     HALF CONE ANGLE.  IF THE TARGET VECTOR LIES WITHIN ZEPS
C              DEGREES OF 180 DEGREE TRANSFER FOR CALL TO INITVI,
C              THEN THE TARGET IS ROTATED INTO PLANE FORMED BY R1-V1.
C     IFLAG    FLAG  IS SET TO 1 IF TARGET VECTOR WAS ROTATED
C              INTO PLANE DUE TO ITS PROXIMITY TO 180
C              DEGREES IN INITVI.  OTHERWISE FLAG IS SET TO 0.
C
C
C **** ENTRY POINT TO INITIALIZE CONIC ROUTINE. MUST BE CALLED ONCE
C     BEFORE USING ANY OF THE CONIC ROUTINES AND THERE AFTER THE ENTRY
C     POINT MUST BE CALLED ONLY IF THE CENTRAL BODY CHANGES
C
```

```
               ENTRY MITINI(ICBODY)
C
C       ESTABLISH CONSTANTS ACCORDING TO CENTRAL BODY
               GO TO(101,102),ICBODY
C       CENTRAL BODY IS MOON
 101           IMOON=1
               RMAX=1.E27
               RMIN=35000.
               RB=5702395.022        @ 1738.09 KM
               XMAX0=1.E16
               GO TO 100
C       CENTRAL BODY IS EARTH
 102           IMOON=0
               RMAX=1.E29
               RMIN=516469.8
               RB=20925738.22        @ 6378.165 KM
               XMAX0=1.E17
               UN(4)=1.
               URT1(4)=1.
               UVT1(4)=1.
               URT2(4)=1.
               UVT2(4)=1.
C       CONVERGENCE CONSTANT FOR KEPLER SUBROUTINE
 100           EPSK=1.E-8
C       CONVERGENCE CONSTANT FOR LAMPERT SUBROUTINE
               EPSL=1.E-8
C       MAX AND MIN VALUES FOR COT(GAM)
               COTMX=31.9843711      @ COT(GAM) FOR GAM=178 DEG 12.5 MIN
               COTMN=-COTMX          @ COT(GAM) FOR GAM= 1 DEG 47.5 MIN
C
               RETURN
C
               DIMENSION R1(4),V1(4),R2(4),V2(4),V1R(4),R2OSET(4)
C
C ****  KEPLER ENTRY POINT
C
               ENTRY KEPMIT(R1,V1,TDES,PMUCB,ZXINIT,R2,V2)
C
```

```
C       IF TDES=0. THEN RETURN INPUT STATE
              IF(ABS(TDES).GT.0.)GO TO 50
              DO 51 K=1,4
              R2(K)=R1(K)
 51           V2(K)=V1(K)
              ZXINIT=0.
              RETURN
 50           CONTINUE
C
C       SET UP COMMON
C
              DO 1 K=1,4
              RT1(K)=R1(K)
 1            VT1(K)=V1(K)
              TD=TDES
              PMU=PMUCB
              SQRPMU=SQRT(PMU)
              XINIT=ZXINIT
C
              CALL KEPMII
C
              DO 2 K=1,4
              R2(K)=RT2(K)
 2            V2(K)=VT2(K)
              ZXINIT=XINIT
C
              RETURN
C
C **** LAMBERT ENTRY POINT
C
              ENTRY LAMMIT(R1,V1,R2,TDES,PMUCB,GCOTG,V1R,V2)
C
              PMU=PMUCB
              SQRPMU=SQRT(PMU)
              DO 3 K=1,4
              RT1(K)=R1(K)
 3            RT2(K)=R2(K)
              TD21=TDES
C
C       COMPUTE VALUE OF SG FROM INPUT RT1,VT1, AND RT2.
C       NOTE--V1 IS USED ONLY TO DETERMINE DIRECTION OF MOTION
C       HERE. THUS IT DOES NOT HAVE TO BE CORRECT IN
C       MAGNITUDE.
C
              UN(1)=VCROSU(RT1,V1)
              GV1(1)=VCROSU(RT1,RT2)
              SG=1.
              IF(VDOT(UN,GV1).LT.0.)SG=-1.
```

```
C       IF INITIAL GUESS OF COTGAM IS AVAILABLE
C       THEN CLEAR IF1 FLAG
              COTGAM=GCOTG
              IF1=1
              IF(ABS(COTGAM).GT.0.)IF1=0
              CALL LAMMII
              DO 8 K=1,4
              V1R(K)=VT1(K)
 8            V2(K)=VT2(K)
              GCOTG=COTGAM
              IF(IF5.EQ.0)RETURN
              WRITE(6,27)
 27           FORMAT( ' LAMMIT HAS FAILED TO FIND A SOLUTION ' )
              CALL MITPNT
              RETURN
C
C **** TIME-THETA ENTRY POINT
C
              ENTRY TTHETA(R1,V1,ZTHETA,MIF6,PMUCB,ZALPN,ZT21,R2,V2)
C
              THETA=ZTHETA
              STHETA=SIN(THETA*DTOR)
              CTHETA=COS(THETA*DTOR)
              IF6=MIF6
              DO 4 K=1,4
              RT1(K)=R1(K)
 4            VT1(K)=V1(K)
              PMU=PMUCB
              SQRPMU=SQRT(PMU)
C
              IF(ABS(1.-CTHETA).GT.0.)GO TO 603
C       THETA =0. PSEUDO COMPUTATIONS
              CALL PARAMI
              ZALPN=ALPN
              ZT21=0.
              DO 20 K=1,4
              R2(K)=RT1(K)
 20           V2(K)=VT1(K)
              RETURN
 603          CALL TTHETI
C
              ZALPN=ALPN
              ZT21=T21
              IF(IF6.EQ.1)RETURN
              DO 5 K=1,4
              R2(K)=RT2(K)
 5            V2(K)=VT2(K)
              IF(IFCOGA.EQ.0.AND.IF7.EQ.0)RETURN
              WRITE(6,25)
 25           FORMAT(' TTHETA FAILED TO FIND A SOLUTION')
              CALL MITPNT
              RETURN
C **** PERAPO ENTRY POINT
C
```

```
                  ENTRY PERAPO(R1,V1,PMUCB,ZHP,ZPN,ZECC)
                  DO 6 K=1,4
                  RT1(K)=R1(K)
 6                VT1(K)=V1(K)
                  PMU=PMUCB
                  SQRPMU=SQRT(PMU)
C
                  CALL PERAPI
C
                  ZHP=HP
                  ZPN=PN
                  ZECC=ECC
                  RETURN
C
C **** TRAD ENTRY POINT
C
                  ENTRY TRAD(R1,V1,R2G,PMUCB,ZSRR,ZT21,R2,V2)
C
                  DO 10 K=1,4
                  RT1(K)=R1(K)
 10               VT1(K)=V1(K)
                  RT2(4)=R2G
                  IF6=0
                  SRR=ZSRR
                  PMU=PMUCB
                  SQRPMU=SQRT(PMU)
                  CALL TRADI
                  ZT21=T21
                  DO 11 K=1,4
                  R2(K)=RT2(K)
 11               V2(K)=VT2(K)
                  IF(IFCOGA.EQ.0.AND.IF9.EQ.0)RETURN
                  WRITE(6,26)
 26               FORMAT( ' TRAD HAS FAILED TO FIND A SOLUTION ' )
                  CALL MITPNT
                  RETURN
C
C **** INITV ENTRY POINT
C
```

```
      ENTRY INITV (R1,V1,T1,IVC,TDES,R2,NOSETZ,ZEPS,PMUCB,V1R,
     . R2OSET,IFLAG)
C
      EPSINV=ZEPS*DTOR
      IF1=1                  $ INITIAL GUESS OF COTGAM NOT AVAILABLE
      PMU=PMUCB
      SQRPMU=SQRT(PMU)
      ISTATE=IVC
      N1=NOSETZ
      TIME1=T1
      TD21=TDES-TIME1
      DO 7 K=1,4
      RT1(K)=R1(K)
      VT1(K)=V1(K)
      RT2(K)=R2(K)
    7 RTT2(K)=R2(K)
      UN(1)=VCROSU(RT1,V1)
C
      CALL INITVI
C
      DO 9 K=1,4
      V1R(K)=VT1(K)
    9 R2OSET(K)=RT2(K)
      IFLAG=IF2
      RETURN
      END
```

APPENDIX E

SUBROUTINE MITPNT

```
      SUBROUTINE MITPNT
C     SUBROUTINE TO PRINT COMMON BLOCK FOR MIT CONIC SUBROUTINES
C
      INCLUDE QCONIC
      INCLUDE QCON
      GAM=0.
      F=0.
      THETA=0.
      IF(ABS(COSF).GT.0..OR.ABS(SINF).GT.0.)F
     .    =ATAN2(SINF,COSF)*RTOD
      IF(ABS(COSGAM).GT.0..OR.ABS(SINGAM).GT.0.)
     .     GAM=ATAN2(SINGAM,COSGAM)*RTOD
      IF(ABS(CTHETA).GT.0..OR.ABS(STHETA).GT.0.)
     .     THETA=ATAN2STHETA,CTHETA)*RTOD
  NAMELIST/NMITSR/A,                          ALP,                ALPN,
 . C1,                C2,                C3,                CK,
 . COEFGX,            COSF,              COSF2,             COSGAM,
 . COTGAM,            COTMIN,            COTMN,             COTMAX,
 . COTMX,             COTTO2,            CTHETA,            CZTA,
 . DCOTG,             DELX,              ECC,
 . EPSINV,            EPSK,              EPSKEP,            EPSL,
 . EPSLAM,            EVEC,              F,                 HA,
 . HP,                GAM,               GV1,               GV2,
 . IDELT,             IFCOGA,            IF1,               IF2,
 . IF3,               IF4,               IF5,               IF6,
 . IF7,               IF8,               IF9,               IFN1,
 . IFW,               IMOON,             IPKEP,             IPLAM,
 . IPTKEP,            IPTLAM,            ITKEP,             ITLAM,
 . ISTATE,            MITKEP,            MITLAM,            N1,
 . N2,                OMEGA,             P1,                P2,
 . PMU,               PN,                RA,                RB,
 . RMAX,              RP,                RT1,               RT2,
 . RTT2P,             RTT2,              SG,                SINF,
 . SINGAM,            SQRPMU,            SQRPN,             STHETA,
 . SZTA,              T21,               T21P,              TD,
 . TD21,              TERR,              THETA,             TIME1,
 . TP,                TR,                TS1,               TS2,
 . TS3,               TS4,               TSKEP,             UEVEC,
 . UN,                URT1,              URT2,              UVT1,
 . UVT2,              VT1,               VT2,               VTT2P,
 . W1,                W2,                W3,                X,
 . XINIT,             XMAX,              XMAXO,             XMIN,
 . XN,                XP,                XR,                X2,
 . X3,                X2CZTA,            ZLAM,              ZTA
      WRITE(6,NMITSR)
      RETURN
      END
```

APPENDIX F

SUBROUTINE APSIDI

```
              SUBROUTINE APSIDI
C
C ****  APSIDES SUBROUTINE (APSIDE)
C       INPUT   RT1,VT1,PMU
C       OUTPUT  RP,RA,ECC
C
              INCLUDE QCONIC
C
C
              CALL PARAMI   @ CONIC PARAMETER SUBROUTINE
              TW1=1.-ALPN*PN
              ECC=0.
              IF(TW1.GT.0.)ECC=SQRT(TW1)
              RP=PN*RT1(4)/(1.+ECC)
              RA=2.*RT1(4)/ALPN-RP
C       RA IS NEG FOR HYPERBOLIC TRAJECTORY OR LARGE FOR HIGHLY
C       ELLIPTIC, PARABOLIC OR SLIGHTLY HYPERBOLIC TRAJECTORY. SET RA=RMAX
              IF(RA.LT.0. .OR. RA.GT.RMAX)RA=RMAX
C
              RETURN
C
        END
```

APPENDIX G

SUBROUTINE DELTII

```
                SUBROUTINE DELTII
C ****  BATTIN'S TRANSCENDENTAL FUNCTIONS (DELTIM)
C       INPUT   C1,C2,X,ZTA,X2,RT1,PMU
C       OUTPUT  T21,SZTA,CZTA,X2CZTA
C
                INCLUDE QCONIC
C       COEFFICIENTS COMPUTED BY HAND FROM STORED VALUE AND SCALE FACTOR
C               DATA CCOEF/ 0.5,  -0.41666678E-1,  0.13888883E-2,
C     .         -0.24801478E-4,  0.27557575E-6,  -0.20879193E-8,
C     .         0.11466301E-10,  -0.47591756E-13,  0.15952475E-15,
C     .         -0.47021409E-18/
C               DATA SCOEF/ 0.16666668,  -0.83333339E-2,   0.19841267E-3,
C     .         -0.27557272E-5,  0.25052219E-7,  -0.16060090E-9,
C     .         0.76452305E-12,  -0.28027516E-14,  0.83655181E-17,
C     .         -0.22099544E-19/
C       EQUATION VALUE ACCORDING TO TRW (SEE BELOW)
C               DATA CCOEF/  0.50000016,  -0.041666680,    1.38888833E-3,
C     1         -2.48014777E-5,             2.75575727E-7, -2.08791932E-9,
C     2          1.14663008E-11,           -4.75917586E-14, 1.59524745E-16
C     3         ,-4.70214090E-19/
C               DATA SCOEF/  1.66666668E-1,-8.33333387E-3, 1.98412673E-4,
C     1         -2.75572720E-6 ,             2.50522187E-8,-1.60600899E-10
C     2         , 7.64523051E-13,           -2.80275162E-15,8.36551806E-18
C     3         ,-2.20995444E-20/
C       THEORETICAL VALUES OF COEFFICIENT ACCORDING TO SUNDANCE PROGRAM,
C       TRW,NAS9-4810,9 SEPTEMBER 1968
                DATA CCOEF/0.5,     -0.041666667,      1.38888889E-3,
      1         -2.48015873E-5,      2.75573192E-7,   -2.08767570E-9,
      2          1.14707456E-11,    -4.77947733E-14,    1.56192070E-16,
      3         -4.11031762E-19/
                DATA SCOEF/ 0.166666667,  -8.33333333E-3, 1.98412698E-4,
      1         -2.75573192E-6,             2.50521084E-8,-1.60590438E-10,
      2          7.64716373E-13,           -2.81145725E-15,8.22063525E-18,
      3         -1.95729411E-20/
                SZTA=0.
                CZTA=0.
                DO 18 IDELT=10,2,-1
                CZTA=(CZTA+CCOEF(IDELT))*ZTA
   18           SZTA=(SZTA+SCOEF(IDELT))*ZTA
                CZTA=CZTA+CCOEF(1)
                SZTA=SZTA+SCOEF(1)
C
                X2=X*X
                X2CZTA=X2*CZTA
                T21=(C1*X2CZTA+X*(C2*X2*SZTA+RT1(4)))/SQRPMU
C
                RETURN
      END
```

APPENDIX H

SUBROUTINE GETXI

```
                SUBROUTINE GETXI
C **** UNIVERSAL VARIABLE SUBROUTINE (GETX)
C       INPUT   STHETA,CTHETA,COTGAM,RT1,ALPN,PN
C       OUTPUT  X,ZTA,C1,C2,X2,IF7
C
                INCLUDE @CONIC
                INCLUDE @CON
C
                DATA COEFGX/-0.333333540,0.200000784,-0.142802172,
     .          0.111006584,-.094528196,0.081388408/
                IFW=0           @ USED ONLY IN GETX - =1,360 DEG TRANSFER
                SQRPN=SQRT(PN)
                COTTO2=STHETA/(1.-CTHETA)       @ COT(THETA/2.)
                W1=(COTTO2-COTGAM)*SQRPN
                IF(ABS(COTTO2).GE.32.)GO TO 360    @ ABS(THETA).LE.3 DEG 35 MIN
                IF(ABS(W1).GE.32.)GO TO 360
C
                DO 362 ITGETX=1,3
                TS1=ALPN+W1*W1
                IF(TS1.LT.0.)GO TO 361   @ CLOSURE THRU INFINITY REQD
                W1=W1+SQRT(TS1)
                IF(ABS(W1).GE.32.)GO TO 360
 362            CONTINUE
C
                A=1./W1
                IF(ABS(A).GE.4.)GO TO 361     @ CLOSURE THRU INFINITY REQD
                GO TO 364
C       W1 OVERFLOW - CALCULATE A USING RECIPROCAL FORMULA
 360            CONTINUE
                IF(W1.LT.0..OR.COTTO2.LT.0.)IFW=1
                W2=ABS(STHETA/(SQRPN*(1.+CTHETA-STHETA*COTGAM)))
                TS2=W2*W2
                W3=1.
                DO 370 ITGETX=1,3
                TS1=ALPN*TS2+W3*W3
                IF(TS1.LT.0.)GO TO 361   @ CLOSURE THRU INFINITY REQUIRED
                W3=SQRT(TS1)+W3
 370            CONTINUE
                A=W2/W3
C
 364            CONTINUE        @ NOW EVALUATE XN
                IF(A.LT.0.)GO TO 361     @ CLOSURE THRU INFINITY REQUIRED
                TS1=ALPN*A*A
                XN=0.
                DO 371 ITGETX=6,1,-1
                XN=(XN+COEFGX(ITGETX))*TS1

 371            CONTINUE
                XN=XN+1.
                XN=16.*A*XN
```

```
C
             IF(IFW.EQ.0)GO TO 372   Q =1,THETA NEAR 360 DEG
             IF(ALPN.LT.0)GO TO 361  Q CLOSURE THRU INFINITY REQD
             XN=TWOPI/SQRT(ALPN)-XN  QSUBTRACT XN FROM 360 DEGREES
 372         CONTINUE
C
             ZTA=XN**2*ALPN
             X=XN*SQRT(RT1(4))
             X2=X*X
             C1=SQRT(PN*RT1(4))*COTGAM
             C2=1.-ALPN
             IF7=0
C
             RETURN
C
C      CLOSURE THRU INFINITY REQD - NO SOLUTION EXISTS
C
 361         CONTINUE
             IF7=1
             RETURN
       END
```

APPENDIX I

SUBROUTINE INITVI

```
              SUBROUTINE INITVI
C       INPUT RT1,VT1,RTT(2),TD,N1,EPS,IF1,GCOTG,PMU
C       OUTPUT RT1,VTT1,RT2,VTT2,RTT2P,COTGAM,IF2
              INCLUDE @CONIC
              INCLUDE @SPNT
C
              OMEGA=COS(EPSINV)
              N2=-1
              Z=0.
              DO 1 K=1,3
              URT2(K)=RT2(K)/RT2(4)
              URT1(K)=RT1(K)/RT1(4)
 1            Z=Z+URT1(K)*URT2(K)
              UN(1)=VCROSU(URT1,VT1)
              IF2=0
 506          IF(Z+OMEGA.GT.0.)GO TO 500 @RTT2 LIES OUTSIDE THE CONE
              IF2=1
              TS1=RT2(1)*UN(1)+RT2(2)*UN(2)+RT2(3)*UN(3)
              GV1(1)=UN(1)*TS1
              GV1(2)=UN(2)*TS1
              GV1(3)=UN(3)*TS1
              GV2(1)=VSUBU(RT2,GV1)
              RT2(1)=GV2(1)*RT2(4)
              RT2(2)=GV2(2)*RT2(4)
              RT2(3)=GV2(3)*RT2(4)
              IF(N2.NE.-1)GO TO 500
              RTT2(1)=RT2(1)
              RTT2(2)=RT2(2)
              RTT2(3)=RT2(3)
 500          GV1(1)=VCROSU(PT1,RT2)
              SG=1.
              IF(VDOT(UN,GV1).LT.0.)SG=-1.
              CALL LAMMII
              IF1=0                  @GCOTG IS NOW AVAILABLE
              IF(N1.EQ.0)RETURN      @PERFORM SINGLE CONIC UPDATI-ONLY
              CALL EADVNC(TIME1,RT1,TD21+TIME1,RTT2P,ISTATE)
              N2=N2+1
              IF(N2.EQ.N1)RETURN   @HAVE INTEGRATED N1+1 TIMES
              RT2(1)=RT2(1)-RTT2P(1)+RTT2(1)
              RT2(2)=RT2(2)-RTT2P(2)+RTT2(2)
              RT2(3)=RT2(3)-RTT2P(3)+RTT2(3)
              RT2(4)=SQRT(RT2(1)*RT2(1)+RT2(2)*RT2(2)+RT2(3)*RT2(3))
              GO TO 506
              END
```

APPENDIX J

SUBROUTINE KEPMII

```
              SUBROUTINE KEPMII
C **** KEPLER SUBROUTNE (KEPMIT)
C       INPUT   RT1,VT1,TD,XINIT,XP,T21P
C       OUTPUT  RT2,VT2,T21,X
C
              INCLUDE QCONIC
              INCLUDE QCON
              INCLUDE QSPNT
C       IF XINIT IS NON ZERO THEN DO NOT RESET XP AND T21P.
C       OTHERWISE ZERO THESE QUANTITIES
              IF(ABS(XINIT).GT.0)GO TO 300
              XP=0.
              T21P=0.
 300          CONTINUE
C
C       SAVE INPUT VARIABLES WHICH ARE CHANGED DURING THE ITERATION LOOP
              TSKEP(1)=XINIT
              TSKEP(2)=XP
              TSKEP(3)=T21P
              TSKEP(4)=TD
              IPTKEP=0       @ =0,DO NOT PRINT EACH ITERATION
              IPKEP=JPKEP
C       PRINT OUT CALL LIST IF JPKEP IS GREATER THAN 0
              IF(JPKEP.EQ.0)GO TO 201
              WRITE(6,200)
 200          FORMAT( ' * * * * KEPMIT IS CALLED WITH THE FOLLOWING '
     .            'VALUES * * * * * ')
              NAMELIST/NLK4/RT1,VT1,TD,T21P,XINIT,X,XP,XR,TR
              WRITE(6,NLK4)
 201          CONTINUE
 113          CONTINUE
              MITKEP=20      @MAX NUMBER ITERATIONS ALLOWED
              ITKEP=0    @ITERATION COUNTER
              XR=0.
              TR=0.
              C1=0.
              C2=0.
              DO 2 K=1,3
              URT1(K)=RT1(K)/RT1(4)
              C1=C1+RT1(K)*VT1(K)
 2            C2=C2+VT1(K)*VT1(K)
              C1=C1/SQRPMU
              C2=C2*RT1(4)/PMU-1.
              ALP=(1.-C2)/RT1(4)  @ ALP.LT.0. - HYPERBOLA, ELSE ELLIPSE
C
              XMAX=XMAXO
              IF(ABS(ALP).LT.1.E-30)GO TO 1
              IF(ALP.LT.0.)XMAX=SQRT(50./(-ALP))
              IF(ALP.GT.0.)XMAX=TWOPI/SQRT(ALP)
              IF(XMAX.GT.XMAXO)XMAX=XMAXO
 1            CONTINUE
C
              IF(TD.LT.0.)GO TO 101    @ YES - NEGATIVE TRANSFER TIME
              TP=XMAX/(ALP*SQRPMU)     @ TP - ORBITAL PERIOD
C
```

```
            IF(TP.LT.0.)GO TO 102
C     IF POSITIVE TRANSFER TIME AND POSITIVE ORBITAL PERIOD REDUCE TD
C     UNTIL 0.LT.TD.LT.TP
 103        IF(TD.LT.TP)GO TO 102    @ FORCE 0.LE.TD.LT.TP
            TD=TD-TP
            XR=XR+XMAX
            TR=TR+TP
            GO TO 103
C
 102        CONTINUE
            X=XINIT-XR
            XMIN=0.
            IF(X.LE.0. .OR. X.GE.XMAX)X=XMAX/2.
            GO TO 104
 101        CONTINUE @ NEGATIVE TRANSFER TIME
            XMIN=-XMAX
            XMAX=0.
            X=XINIT
            IF(X.GE.0. .OR. X.LT.XMIN)X=XMIN/2.
C
 104        CONTINUE      @ BRANCHES OF TEST OF TD COME TOGETHER HERE
            IF(ABS(T21P).GT.0.)T21P=T21P-TR
            DELX=X
            IF(ABS(XP).GT.0.)DELX=X-XP+XR
            EPSKEP=ABS(EPSK*TD)
C
C     SET INDICATORS FOR ITERATOR SUBROUTINE. C1,C2 AND ALP ARE
C     CONSTANT WITHIN THE LOOP
C
            IF4=0
            IF3=0
            CK=0.
            NAMELIST/NLK3/ITKEP,TD,T21,T21P,TERR,EPSKEP,XINIT,X,XP,
     .      DELX,EPSK,XMAX,XMIN,C1,C2,ALP,ZTA,CZTA,SZTA
C
C     START OF ITERATION LOOP
C
 105        CONTINUE
            X2=X*X
            ZTA=ALP*X2
C
            CALL DELTII   @ BATTIN'S TRANSCENDENTAL FUNCTIONS
C
            TERR=TD-T21
            IF(ABS(TERR).LE.EPSKEP)GO TO 106 @HAS CONVERGED
C
```

```
C       IS PRINTING OF ITERATION REQUIRED?
                IF(IPTKEP.EQ.0.AND.JPKEP.EQ.0)GO TO 115
C       PRINTING IS REQUIRED - CHECK IF NORMAL PRINTING
                IF(IPTKEP.EQ.0)GO TO 116
C       TROUBLE PRINTING
                IF(IPKEP.LT.JPTKEP)GO TO 115
                IPKEP=0
                WRITE(6,NLK3)
                GO TO 115
C       NORMAL PRINTING
 116            IF(IPKEP.LT.JPKEP)GO TO 117
                IPKEP=0
                WRITE(6,NLK3)
 117            CONTINUE
C       CONTINUE ITERATIONS AS REQUIRED
 115            IPKEP=IPKEP+1
C       CONTINUE IF MAXIMUM NUMBER OF
C       ITERATIONS HAS NOT BEEN EXCEEDED
                IF(ITKEP.LE.MITKEP)GO TO 120
C       KEPLER HAS NOT CONVERGED WITHIN ALLOWABLE NUMBER OF ITERATIONS
                IF(IPTKEP.EQ.1)GO TO 106
                WRITE(6,112)
 112            FORMAT(// ' * * *  KEPMIT DID NOT CONVERGE WITHIN'
     .          ' MAXIMUM  NUMBER OF ITERATIONS  * * * * * '/)
C       GO BACK AND PRINT ITERATIONS IF REQUIRED
                IPKEP=JPTKEP
                IPTKEP=1
                XINIT=TSKEP(1)
                XP=TSKEP(2)
                T21P=TSKEP(3)
                TD=TSKEP(4)
                WRITE(6,200)
                WRITE(6,NLK4)
                IF(JPTKEP.GT.0)GO TO 113
                GO TO 106
C       CALL ITERATOR (PSEUDO CALL)
C
 120            IF(ABS(T21-T21P).GT.0.)GO TO 121
                DELX=0.
                GO TO 110
 121            DELX=DELX*TERR/(T21-T21P)
                IF(DELX.GT.0.)GO TO 107
                XMAX=X         @ MOVE IN UPPER BOUND
                IF(XMIN.GE.(X+DELX))DELX=0.9*(XMIN-X)
                GO TO 108
C
```

```
 107            CONTINUE
                XMIN=X          @ MOVE IN LOWER BOUND
                IF(XMAX.LT.(X+DELX))DELX=0.9*(XMAX-X)
C
 108            CONTINUE
C
C      DECREMENT ITERATION COUNTER
                ITKEP=ITKEP+1
C      INCREMENT X BY DELX AND CONTINUE ITERATION.
C      IF DELX IS TOO SMALL TO EFFECT X THEN LEAVE LOOP
                TS1=X
                X=X+DELX
                T21P=T21
                IF(ABS(X-TS1).GT.0.)GO TO 105
C
C      ITERATION HAS NOT CONVERGED, BUT DELX IS SO SMALL IT WILL NOT
C      EFFECT X
 110            CONTINUE
                IF(JPKEP+IPTKEP).EQ.0)GO TO 106
                WRITE(6,111)
 111            FORMAT(//1H , ' * * * * DELX IS TOO SMALL TO EFFECT X * *
     . * * * ')
                WRITE(6,NLK3)
C
C      THROUGH ITERATING, CALL STATE VECTOR SUBROUTINE
C
 106            CALL NEWSTI
C      SET XP AND T21P TO SOLUTION VALUE
                XINIT=XP
                XP=X+XR
                T21P=T21+TR
                IF(IPTKEP.EQ.0.AND.JPKEP.EQ.0)RETURN
                WRITE(6,NLK3)
                WRITE(6,114)
                NAMELIST/NLK1/X,TD,T21,TFRR,RT2,VT2,ITKEP
                WRITE(6,NLK1)
 114            FORMAT(' * * * * * SOLUTION OBTAINED BY KEPMIT IS * * * '
     .          ' * * ')
C
                RETURN
C
C
       END
```

APPENDIX K

SUBROUTINE LAMMII

```
               SUBROUTINE LAMMII
C  **** LAMBERT SUBROUTINE (LAMMIT)
C       INPUT   RT1,RT2,TD21,SG,IF1,DCOTG,IF2,UN,PMU,IFN1
C       OUTPUT  VT1,VT2,COTGAM,IF5
C
               INCLUDE QCONIC
C
               INCLUDE GSPNT
               IF5=0                 @ CLEAR SOLUTION FLAG
               IF3=1                 @ SET FLAG FOR ITERATOR
               IPLAM=JPLAM
C       PRINT OUT CALL LIST IF JPLAM.GT.0
               IF(JPLAM.EQ.0)GO TO 201
               WRITE(6,200)
 200           FORMAT(' LAMMIT IS CALLED WITH THE FOLLOWING VALUES')
               NAMELIST/NLK4/RT1,RT2,TD21,SG,IF1,DCOTG,COTGAM
               WRITE(6,NLK4)
 201           CONTINUE
 113           CONTINUE
               MITLAM=20  @ MAXIMUM NUMBER OF ITERATIONS ALLOWED
               ITLAM=0
C              IPTLAM=0
C       PSEUDO CALL TO GEOMETRIC PARAMETER SUBROUTINE
C
               CTHETA=0.
               DO 1 K=1,3
               URT1(K)=RT1(K)/RT1(4)
               URT2(K)=RT2(K)/RT2(4)
 1             CTHETA=CTHETA+URT1(K)*URT2(K)
               UN(1)=VCROSM(URT1,URT2)
               STHETA=UN(4)*SG
               UN(1)=UN(1)/STHETA
               UN(2)=UN(2)/STHETA
               UN(3)=UN(3)/STHETA
               UN(4)=1.
C       RESUME
               ZLAM=RT1(4)/RT2(4)
               EPSLAM=ABS(EPSL*TD21)
               P1=1.-CTHETA
               IF(P1.LT.1.E-8)GO TO 360 @TRANSFER TOO NEAR 180 OR 360
               P2=CTHETA-ZLAM
               COTMAX=STHETA/P1+SQRT(2.*ZLAM/P1)
C       TEST FOR GAM.LT.(1 DEG 47.5 MIN) - IF SO LIMIT COTMAX
               IF(COTMAX.GT.COTMX)COTMAX=COTMX
               COTMIN=COTMN
               IF(SG.GT.0.)COTMIN=P2/STHETA
C       TEST FOR GAM.GT.(178 DEG 12.5 MIN) - IF SO LIMIT COTMIN
               IF(COTMIN.LT.COTMN)COTMIN=COTMN
C
```

```
C       COMPUTE INITIAL GUESS OF COTGAM IF IF1 IS SET - IF1 IS NORMALLY
C       SET ONLY ON THE FIRST CALL TO LAMBERT
                CK=1.E-5
                IF(IF1.EQ.0)GO TO 339
                CK=0.25
                COTGAM=(COTMAX+COTMIN)/2.     @ INITIAL VALUE FOR COTGAM
                DCOTG=COTGAM          @ INITIAL VALUE OF  DCOTG
                T21P=0.
                NAMELIST/NLK3/ITLAM,COTGAM,DCOTG,EPSLAM,COTMAX,COTMIN,
       .        TD21,T21P,T21,TERR,CTHETA,STHETA,P1,P2,CK,ZLAM,PN,ALPN,
       .        ZTA,CZTA,SZTA,X,IF7
 339            IF(JPLAM.GT.0.OR.IPTLAM.GT.0)WRITE(6,NLK3)
C
C       START OF ITERATION LOOP
 302            CONTINUE                      @ LAMBLOOP ENTRY POINT
C
                PN=P1/(COTGAM*STHETA-P2)
                IF(PN.LE.0.)GO TO 303    @ CORRECTIVE ACTION REQUIRED
                ALPN=2.-PN*(1.+COTGAM**2)
                CALL GETXI
                T21P=T21
                IF(IF7.EQ.1)GO TO 303    @ CORRECTIVE ACTION REQD
                CALL DELTII   @ CALCULATE TRANSFER TIME T21
C
                TERR=TD21-T21
                IF(ABS(TERR).LT.EPSLAM)GO TO 305 @HAS CONVERGED
C       INCREMENT ITERATION COUNTER
                ITLAM=ITLAM+1
C       IS PRINTING OF ITERATIONS REQUIRED?
                IF(IPTLAM.EQ.0.AND.JPLAM.EQ.0)GO TO 115
C       PRINTING IS REQUIRED - CHECK IF NORMAL PRINTING
                IF(IPTLAM.EQ.0)GO TO 116
C       TROUBLE PRINTING
                IF(IPLAM.LT.JPLAM)GO TO 115
                IPLAM=0
                WRITE(6,NLK3)
                GO TO 115
C       NORMAL PRINTING
 116            IF(IPLAM.LT.JPLAM)GO TO 117
                IPLAM=0
                WRITE(6,NLK3)
 117            CONTINUE
C       CONTINUE ITERATIONS AS REQUIRED
 115            IPLAM=IPLAM+1
                IF(ITLAM.GT.MITLAM)GO TO 332
C
C       PSEUDO CALL TO ITERATOR SUBROUTINE  (WITH IF4=0)
C
```

```
           IF(IF3.EQ.0)GO TO 320  @ FALL THRU ONLY ON FIRST ITERATION
           IF3=0
           DCOTG=CK*(COTMAX-COTMIN)
           DCOTG=SIGN(DCOTG,TERR)
           GO TO 321
 320       CONTINUE                @ BRANCH USED AFTER FIRST PASS
           DCOTG=DCOTG*TERR/(T21-T21P)
 321       CONTINUE
           IF(DCOTG.GT.0.)GO TO 322
           COTMAX=COTGAM       @ LOWER UPPER BOUND
           IF(COTMIN.GT.COTGAM+DCOTG)DCOTG=0.9*(COTMIN-COTGAM)
           GO TO 323
 322       COTMIN=COTGAM       @ RAISE LOWER BOUND
           IF(COTMAX.LT.COTGAM+DCOTG)DCOTG=0.9*(COTMAX-COTGAM)
 323       CONTINUE            @ RETURN FROM ITERATOR SUBROUTINE
C
C     CHANGE COTGAM AND GO TO START OF ITERATION LOOP.
C     IF DCOTG DOES NOT CHANGE COTGAM THEN LEAVE LOOP
           TS1=COTGAM
           COTGAM=COTGAM+DCOTG
           IF(ABS(COTGAM-TS1).GT.0.)GO TO302
           GO TO 331
C     IF((PN.LE.0.) .OR.IF7.EQ.1)THEN JUMP HERE FOR CORRECTIVE ACTION
 303       CONTINUE             @ NEGP
           IF(OPTLAM.EQ.0) GO TO 400
           WRITE(6,401)
 401       FORMAT(/' CORRECTIVE ACTION REQUIRED IN LAMMIT')
           WRITE(6,NLK3)
 400       CONTINUE
           IF(DCOTG.LT.0.)GO TO 313
           GO TO 403
```

```
C       IF ALPN BECOMES TOO LARGE THEN COME HERE FOR CORRECTIVE ACTION
 310            CONTINUE                   @ HIENERGY
                IF(OPTLAM.EQ.0) GO TO 403
                WRITE(6,401)
                WRITE(6,NLK3)
 403            CONTINUE
                COTMIN=COTGAM
                GO TO 311
C       IF T21 BECOMES TOO LARGE THEN COME HERE FOR CORRECTIVE ACTION
C312            CONTINUE
C               T21=T21P
 313            CONTINUE                   @ LOENERGY
                COTMAX=COTGAM
 311            CONTINUE
                DCOTG=DCOTG/2.
                TS1=COTGAM
                COTGAM=COTGAM-DCOTG
                IF(ABS(COTGAM-TS1).GT.0.) GO TO 302 @GO TO LAMBLOOP
                GO TO 330
C
 332            WRITE(6,333)
 333            FORMAT(' * * * LAMBERT DID NOT CONVERGE WITHIN EPLSAM'
     .          '  AFTER MAXIMUM ITERATIONS * * * ')
                WRITE(6,NLK3)
                GO TO 330
 331            WRITE(6,337)
 337            FORMAT(' * * * DCOTG IS TOO SMALL * *  * ')
                WRITE(6,NLK3)
                GO TO 330
C
```

```
 360           CONTINUE
C
               WRITE(6,119)
 119           FORMAT( ' ***TRANSFER TOO NEAR 180 OR 360 DEGREES IN '
     .         '  LAMMIT  ***')
C              SET IF5 FLAG
               IF5=1
               WRITE(6,NLK3)
               RETURN
 330           IF(ABS(TERR).LT.EPSLAM)GO TO 305
               WRITE(6,338)
 338           FORMAT(' * * *LAMBERT DID NOT CONVERGE WITHIN'
     .                  ' TOLERANCE OF CK1*TD21')
               IF5=1          @ SET FOR NO SOLUTION
C         LAMBERT HAS NOT CONVERGED WITHIN ALLOWABLE NUMBER OF ITERATIONS
               IF(IPTLAM.EQ.1)GO TO 305
C         GO BACK AND PRINT ITERATIONS AS REQUIRED
               IPLAM=JPTLAM
               IPTLAM=1
               WRITE(6,200)
               WRITE(6,NLK4)
               IF(JPTLAM.GT.0)GO TO 113
 305           CONTINUE       @ CALCULATE VT1
               TS2=SQRT(PN*PMU/RT1(4))
               GV2(1)=VCROSS(UN,URT1)
               VT1(1)=(URT1(1)*COTGAM+GV2(1))*TS2
               VT1(2)=(URT1(2)*COTGAM+GV2(2))*TS2
               VT1(3)=(URT1(3)*COTGAM+GV2(3))*TS2
               VT1(4)=SQRT(VT1(1)*VT1(1)+VT1(2)*VT1(2)+VT1(3)*VT1(3))
C
C
C         GO TO 'NEWSTATE' VIA INTERNAL ENTRY POINT 'LAMENT' TO COMPUTE
C         TERMINAL VELOCITY VT2 IF FLAG IFN1 IS CLEAR
               IF(IFN1.EQ.0)CALL LAMENT
C
               IF(IPTLAM.GT.0.OR.JPLAM.EQ.0)RETURN
               WRITE(6,NLK3)
               WRITE(6,114)
 114           FORMAT('* * * SOLUTION OBTAINED BY LAMMIT IS * * * ')
               NAMELIST/NLK1/RT1,VT1,RT2,VT2,TD21,T21,COTGAM,SG,IF5
               WRITE(6,NLK1)
               RETURN
C
          END
```

APPENDIX L

SUBROUTINE NEWSTI

```
                  SUBROUTINE NEWSTI
C ****  STATE VECTOR SUBROUTINE (NEWST)
C       CALLED BY TTHETA,KEPLER,LAMBERT
C       INPUT     RT1,VT1,URT1,X,ZTA,SZTA,CZTA,X2CZTA,T21,PMU
C       OUTPUT    RT2,VT2
C
                  INCLUDE @CONIC
C
C
                  X2=X*X
                  X3=X2*X
                  X2CZTA=X2*CZTA
                  TS1=RT1(4)-X2CZTA
                  TS2=T21-X3*SZTA/SQRPMU
                  DO 2 K=1,3
                  URT1(K)=RT1(K)/RT1(4)
 2                RT2(K)=URT1(K)*TS1+VT1(K)*TS2
                  RT2(4)=SQRT(RT2(1)*RT2(1)+RT2(2)*RT2(2)+RT2(3)*RT2(3))
C
                  ENTRY LAMENT          @ ENTRY POINT FROM LAMMII
C
 1                TS1=SQRPMU*X*(ZTA*SZTA-1.)/RT2(4)
                  TS2=1.-X2CZTA/RT2(4)
                  VT2(1)=URT1(1)*TS1+VT1(1)*TS2
                  VT2(2)=URT1(2)*TS1+VT1(2)*TS2
                  VT2(3)=URT1(3)*TS1+VT1(3)*TS2
                  VT2(4)=SQRT(VT2(1)*VT2(1)+VT2(2)*VT2(2)+VT2(3)*VT2(3))
C
                  RETURN
        END
```

APPENDIX M

SUBROUTINE PARAMI

```
                SUBROUTINE PARAMI
C
C **** CONIC PARAMETERS SUBROUTINE (PARAM)
C       CALLED BY TIMRAD,TTHETA
C       INPUT   RT1,VT1,PMU
C       OUTPUT  ALPN,PN,COTGAM,UN,URT1,IFCOGA
C
                INCLUDE QCONIC
C
C
                SG=1.           @ FORCES GAM TO BE CALCULATED IN RANGE
                                @       (0,180) DEG
                IF2=0           @ FORCES GEOM TO CALCULATE UN
                IFCOGA=0        @ CLEARS COTGAM OVERFLOW INDICATOR
C
C       PSEUDO CALL TO GEOMETRIC PARAMETERS SUBROUTINE
C       INPUT RT1,VT1,IF2,SG
C       OUTPUT SINGAM,COSGAM,UN,URT1,UVT1
C
                COSGAM=0.
                DO 1 K=1,3
                URT1(K)=RT1(K)/RT1(4)
                UVT1(K)=VT1(K)/VT1(4)
 1              COSGAM=COSGAM+URT1(K)*UVT1(K)
                GV1(1)=VCROSM(URT1,UVT1)
                SINGAM=GV1(4)
                UN(1)=GV1(1)/SINGAM
                UN(2)=GV1(2)/SINGAM
                UN(3)=GV1(3)/SINGAM
C
C       RESUME
C
                COTGAM=COSGAM/SINGAM
C       IF GAM NOT IN RANGE 1 DEG 47.5 MIN TO 178 DEG 12.5 MIN
C       THEN SET INDICATOR
                IF(ABS(COTGAM).GT.COTMX)IFCOGA=1
                C3=RT1(4)*VT1(4)*VT1(4)/PMU
                ALPN=2.-C3      @ RATIO OF RT1(4) TO SEMIMAJOR AXIS
                PN=C3*SINGAM*SINGAM     @ RATIO OF SEMILATUS LATUS TO RT1(4)
C
                RETURN
C
        END
```

APPENDIX N

SUBROUTINE PERAPI

```
                SUBROUTINE PERAPI
C ****  PERICENTER-APOCENTER SUBROUTINE (PERAPO)
C       COMPUTES THE TWO BODY APOCENTER AND PERICENTER ALTITUDES
C       CALLED BY P30,P37,P32 THRU P35,P72 THRU P75,MANUPARM
C       INPUT RT1,VT1,PMU
C       OUTPUT  HA,HP,ECC,PN,RA,RP
C
C
                INCLUDE @CONIC
C
                CALL APSIDI           @ APSIDES SUBROUTINE
C
                HP=RP-RB
                HA=RA-RB
C
                RETURN
C
        END
```

APPENDIX O

SUBROUTINE TRADI

```
                SUBROUTINE TRADI
C ****  TIMERAD SUBROUTINE (TRAD)
C       INPUT   RT1,VT1,PMU,RT2,SRR,IF6
C       OUTPUT  T21,VT2,IF8,IFCOGA,IF5
C
                INCLUDE QCONIC
C
C
                CALL PARAMI   @ CONIC PARAMETERS SUBROUTINE
                IF(IFCOGA.EQ.0)GO TO 500 @ YES - SOLUTION EXISTS
C       PSEUDO CALL TO TTHETA TO INDICATE NO SOLUTION EXISTS
                RETURN
C
 500            CONTINUE
                TS1=COTGAM*SQRT(PN*(2.-ALPN))
                TS2=1.-ALPN
                EVEC(1)=URT1(1)*TS2-UVT1(1)*TS1
                EVEC(2)=URT1(2)*TS2-UVT1(2)*TS1
                EVEC(3)=URT1(3)*TS2-UVT1(3)*TS1
                EVEC(4)=SQRT(EVEC(1)*EVEC(1)+EVEC(2)*EVEC(2)+EVEC(3)
     *          *EVEC(3))
                UEVEC(1)=EVEC(1)/EVEC(4)
                UEVEC(2)=EVEC(2)/EVEC(4)
                UEVEC(3)=EVEC(3)/EVEC(4)
                UEVEC(4)=1.
                IF(EVEC(4).GE.1./262144..AND.EVEC(4).LT.8.)GO TO 501
C       FAILURE OF ABOVE TEST INDICATES FAILURE
                IF9=1
                RETURN
C
 501            CONTINUE
                COSF=((PN*RT1(4))/RT2(4)-1.)/EVEC(4)
                IF(ABS(COSF).GE.1.)GO TO 503
                COSF2=COSF*COSF
                IF(COSF2.GT.1.)GO TO 503
                IF8=0
                SINF=SRR*SQRT(1.-COSF2)
                GO TO 504
C
 503            CONTINUE      @ ABS(COSF).GE.1.) - SET COSF=1. WITH
                COSF=SIGN(1.,COSF)      @ SAME SIGN
                SINF=0.
                IF8=1           @ INDICATES RT2(4) IS OUTSIDE RANGE
```

```
C
 504          CONTINUE
              CTHETA=0.
              GV2(1)=VCROSU(UN,UEVEC)
              DO 2 K=1,3
              URT2(K)=UEVEC(1)*COSF+GV2(K)*SINF
 2            CTHETA=CTHETA+URT1(K)*URT2(K)
              STHETA=SQRT(1.-CTHETA**2)
              GV1(1)=VCROSU(URT1,URT2)
              GV1(1)=GV1(1)+UN(1)
              GV1(2)=GV1(2)+UN(2)
              GV1(3)=GV1(3)+UN(3)
              GV1(4)=SQRT(GV1(1)*GV1(1)+GV1(2)*GV1(2)+GV1(3)*GV1(3))
              IF(GV1(4).LT.1.)STHETA=-STHETA
C
              CALL GETXI
C
              IF9=0 @INDICATES SOLUTION IS VALID
              CALL DELTII   @ CALCULATE T21
              IF(IF6.EQ.1)RETURN
              CALL NEWSTI   @ CALCULATE FINAL STATE
              RETURN
      END
```

APPENDIX P

SUBROUTINE TTHETI

```
              SUBROUTINE TTHETI
C **** TIME-THETA SUBROUTINE (TTHETA)
C      CALLED BY CSI/A,CDHMVR,P34(AND P74),PREC/TT (IN P35 AND P75),TRAD
C      INPUT   RT1,VT1,PMU,STHETA,CTHETA,IF6
C      OUTPUT  RT2,VT2,IF7,IFCOGA
C
              INCLUDE QCONIC
C
C
              CALL PARAMI          @ CONIC PARAMETER SUBROUTINE
              IF(IFCOGA.EQ.1)GO TO 400       @ NO SOLUTION
              CALL GETXI
              IF(IF7.EQ.1)GO TO 401    @ NO SOLUTION
              IFCOGA=0
              CALL DELTII   @BATTIN'S TRANSCENDENTAL FUNCTIONS
C
              IF(IF6.EQ.1)RETURN @ RETURN T21
              CALL NEWSTI          @ STATE VECTOR SUBROUTINE
              RETURN               @ RETURN T21,RT2,VT2
C
 400          CONTINUE  @ NO SOLUTION - GAM TO NEAR 0 OR 180 DEG
              IFCOGA=1
              RETURN
C
 401          CONTINUE      @ NO SOLUTION - CLOSURE THRU INFINITY
              IFCOGA=0
              RETURN
       END
```

APPENDIX Q

SUBROUTINE DELTII - SERIES SUMMATION FORM

```
                SUBROUTINE DELTII
C **** COMPUTES BATTINS TRANSCENDENTAL FUNCTIONS BY MEANS OF SERIES
C
                INCLUDE @CONIC
C
                CZTA=.5
                SZTA=1./6.
                F2N=2.
                BASE=SZTA
                ICONT=0
 7              F2N=F2N+2.
                BASE=-BASE*ZTA/F2N
                CB=CZTA
                CZTA=CZTA+BASE
C
                BASE=BASE/(F2N+1.)
                SB=SZTA
                SZTA=SZTA+BASE
C
                IF(ABS(SZTA-SB).GT.0.OR.ABS(CZTA-CB).GT.0)GO TO B
                GO TO 9
 8              ICONT=ICONT+1
                IF(ICONT.LT.100)GO TO 7
                WRITE(6,29)
 29             FORMAT(//// ' SERIES FAILED TO CONVERGE IN DELTII ' )
C
 9              X2=X*X
                X2CZTA=X2*CZTA
C
                T21=(C1*X2CZTA+X*(C2*X2*SZTA+RT1(4)))/SGRPMU
                RETURN
          END
```